Technische Universität Berlin
Institut für Mathematik

Prof. Rolf H. Möhring
Felix König

ADM II – Lineare und Ganzzahlige Optimierung
Wintersemester 2008/2009

Max Klimm

1. Programmieraufgabe

Abgabe: Montag, 5.1.2009
Abnahme: während der Vorrangzeit im UNIX-Pool am Dienstag, dem 6.1.2008

Implementiert den revidierten Simplex-Algorithmus aus der Vorlesung in Java oder C/C++. Euer fertiges Programm muss

- `.lp`-Dateien einlesen,
- Lösungen mit Zielfunktionswerten ausgeben und in Dateien schreiben,
- auf Wunsch Informationen wie reduzierte Kosten und in oder aus der Basis wechselnde Variablen je Iteration ausgeben,
- lückenlos und sauber mit JavaDoc, Doxygen oder einem anderen im UNIX-Pool verfügbaren System dokumentiert sein und
- mit Java 1.6 oder gcc/g++ 4.x im Pool kompilierbar sein.

Auf unserer Website werden Testbeispiele zur Verfügung stehen, mit denen euer Code korrekt umgehen muss.

Erstellt am Ende ein Archiv mit einer Datei `README`, die eine kurze Anleitung zum Kompielieren des Codes und Erstellen der Dokumentation enthält und schickt es **spätestens am Montag, dem 5.1.2009** an `klimm@math.tu-berlin.de` und `fkoenig@math.tu-berlin.de`. Weitere Details zur Abgabe werden in der UE besprochen und auf unserer Website veröffentlicht.

Hier noch ein paar Tipps zur Aufgabe, den ihr aber nicht folgen müsst:

- Teilt euch die Arbeit gut ein und setzt euch Termine für Zwischenziele. Drei Wochen erscheinen wie eine lange Zeit, es gibt aber auch viel zu tun. Überraschende Bugs nehmen keine Rücksicht auf Abgabetermine...
- Nehmt euch ausreichend Zeit für das Erstellen eines Implementationskonzepts, bevor ihr die ersten Zeilen Code schreibt.
- Verwendet Subversion.
- Implementiert zuerst alle nötigen Routinen für Matrix- und Vektoroperationen.
- Implementiert dann Phase II des Verfahrens und testet es mit “hard-coded” Testinstanzen.
- Ergänzt dann Phase I und testet auf ähnliche Weise.
- Integriert einen Parser für `.lp`-Dateien. Dank Axel Werner steht ein Java-Parser auf unserer Website zur Verfügung, für dessen Fehlerfreiheit allerdings keine Gewähr übernommen wird...
- Testet einzelne Teile eures Codes gewissenhaft und so früh wie möglich. Je einfacher ein Stück Code, desto leichter findet man Fehler.
- Integriert von Anfang die Möglichkeit, sich den Fortschritt des Programms in unterschiedlichem Detailgrad ausgeben zu lassen. Am Ende spart man so gegenüber einem sich ständig ändernden Dschungel aus einzelnen Ausgaben immer eine Menge Zeit.